ECEN 449: Microprocessor System Design

Lab Report 1

Using the Integrated Software and Environment (ISE)

Name: Peng Li

UIN: 822003660

Date: 9/11/2013

1. Introduction

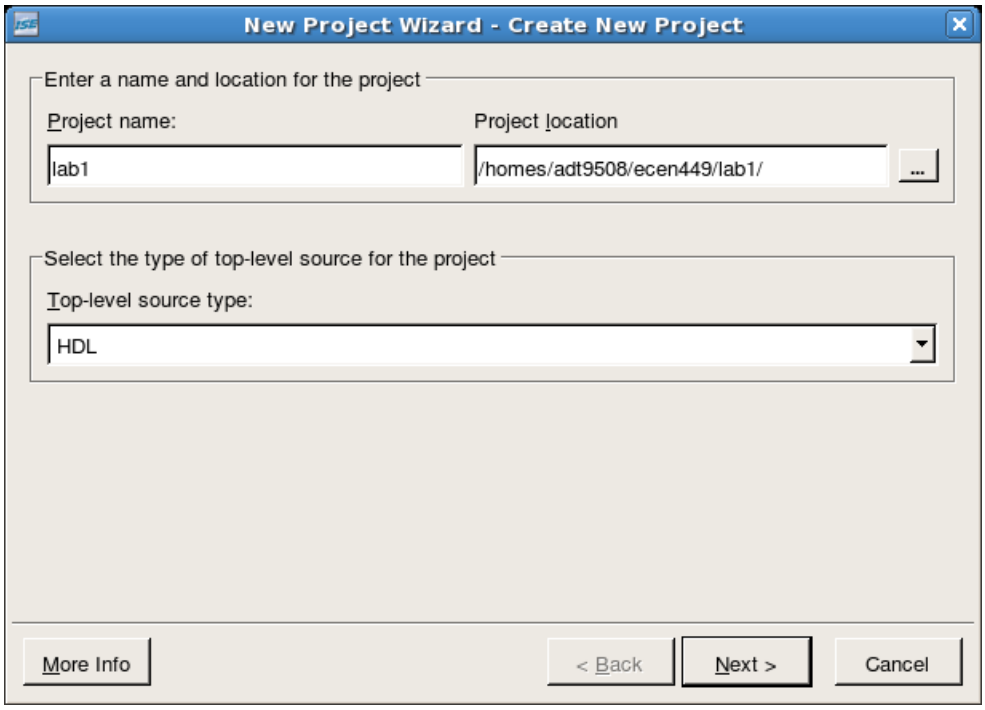
The main purpose of this lab is to get familiar with Xilinx FPGA design tool – Integrated Software Environment (ISE). There are three design projects during the lab. The first project is to create a 4-input and 4- output switch device. The second project is to build up a led counter. And the third project is to design a Jackpot game.

All three projects are implemented on Xilinx XUP board. The model of the development board is Xilinx Virtex5 XC5VLX110T. The synthesis tool is XST. And the program language is Verilog.

2. Procedure

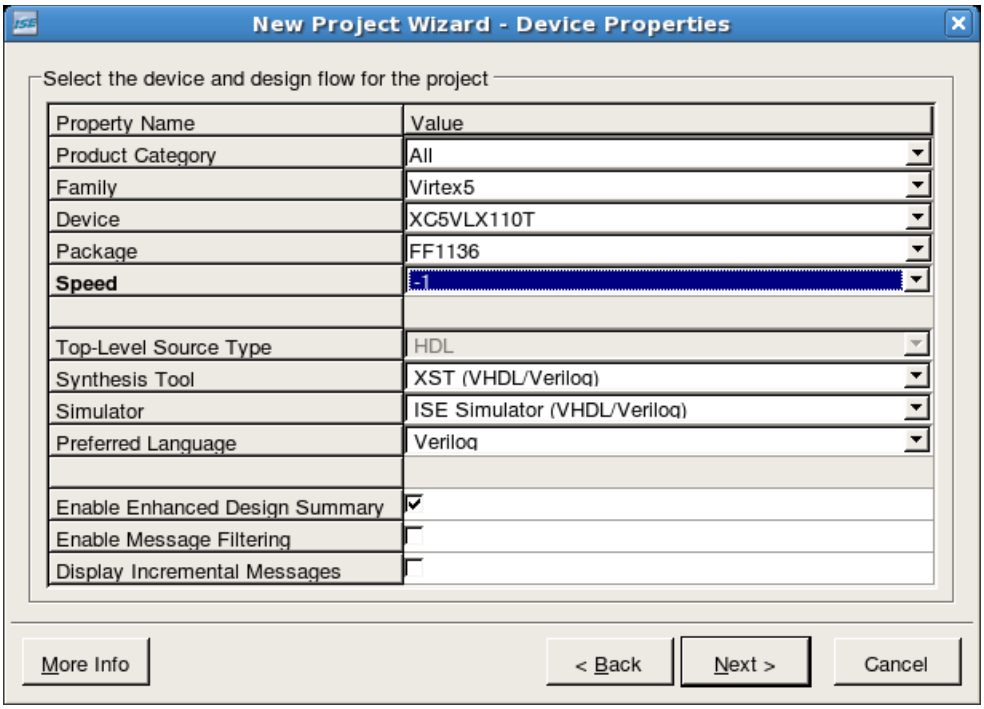
Lab1 a.

1. Open ISE, select File -> New Project. Then the New Project Wizard opens.



The project name is : lab1 and the project location is ../ecen449/lab1/. Chose HDL for the Top-level source type and click Next.

2. The Device Properties should be set as the following figure.



3. Then, Create New Source window shows up. Click on New Source, select Verilog Module for the source type and type in 'switch.v' as the file name.
4. Next the Define Module window shows up. Type in an input port called 'SWITCHES', the bandwidth is 3 down to 0. Type in an output port called 'LEDS', the bandwidth is 3 down to 0. Then click on Next and click on Finish on the Summary window.
5. From the Source window, open 'switch.v' file. Type in the code showed in appendix lab1 a. When the SWITCHES input is high, the corresponding LEDS will light up.
6. Next create a User Constraint File containing the location of the DIP switches and LEDS on the XUP board. The .ucf file is attached in the appendix lab1 a. And the pin assignments are showed in the char below.

Table 1-5: **DIP Switch Connections (SW8)**

SW4	FPGA Pin
GPIO_DIP_SW1	U25
GPIO_DIP_SW2	AG27
GPIO_DIP_SW3	AF25
GPIO_DIP_SW4	AF26
GPIO_DIP_SW5	AE27
GPIO_DIP_SW6	AE26
GPIO_DIP_SW7	AC25
GPIO_DIP_SW8	AC24

Table 1-6: **User and Error LED Connections**

Reference Designator	Label/Definition	Color	FPGA Pin	Buffered
DS20	LED North	Green	AF13	Yes
DS21	LED East	Green	AG23	Yes
DS22	LED South	Green	AG12	Yes
DS23	LED West	Green	AF23	Yes
DS24	LED Center	Green	E8	Yes
DS17	GPIO LED 0	Green	H18	Yes
DS16	GPIO LED 1	Green	L18	Yes
DS15	GPIO LED 2	Green	G15	Yes
DS14	GPIO LED 3	Green	AD26	No
DS13	GPIO LED 4	Green	G16	Yes
DS12	GPIO LED 5	Green	AD25	No
DS11	GPIO LED 6	Green	AD24	No
DS10	GPIO LED 7	Green	AE24	No
DS6	Error 1	Red	F6	No
DS6	Error 2	Red	T10	No

7. In this lab we will use U25, AG27, AF25, AF26 for input switches pins and H18, L18, G15, AD26 for output LEDS pins.
8. Add the .ucf file to the project by right clicking the Sources Window and select 'Add Source'.
9. Next, select 'switch.v' in the 'Sources' window. In the Process window, click on 'Generate Programming File'. And turn on the XUP board.
10. Download the code into the XUP board following the instructions show on the screen and the right click 'xc5vlx110t' in the IMPACT window and select 'Program'.

Lab1 b

1. Counter/ divider
   The clock frequency is 100MHz. So a modified 1Hz clock is needed in the design. A counter could be used to count the clock cycles of the original clock.
   The pseudo code is shown below:

```
@ posedge CLOCK
If (COUNTER == 5e7)      COUNTER = 0;
                         MODIFIED_CLOCK = ~MODIFIED_CLOCK;
```

Else COUNTER = COUNTER + 1;
MODIFIED_CLOCK = MODIFIED_CLOCK;

Hence the frequency of the original clock is divided by 10^8.

2. Button Set
   In the second project, we build a led counter which will count up if the Up-button is pressed, or count down if the Down-button is pressed. And the third input button is the Mid-button, which is used for RESET.

   The pseudo code is shown below:

   If (UP_BUTTON == 1) LEDS = LEDS + 1;

   Else if (DOWN_BUTTON == 1) LEDS = LEDS -1;

   Else if (RESET == 0) LEDS = 0;

   Else LEDS = LEDS;

3. Pin Assignment
   Most part of the second project is same as the first project, except the pin assignment and the Verilog code. The .ucf file and Verilog code is attached in the appendix. The led output still uses the same as in the first project. Only the input port SWITCHES are changed. And the switch button pin assignment chart is below.

Table 1-7: **User Pushbutton Connections**

Reference Designator	Label/Definition	FPGA Pin
SW10	N (GPIO North)	U8
SW11	S (GPIO South)	V8
SW12	E (GPIO East)	AK7
SW13	W (GPIO West)	AJ7
SW14	C (GPIO Center)	AJ6

   We use U8, V8 for up and down switches pin and AJ6 for reset pin.
   Since clock is implemented in the second project. A clock pin should be assignment to the input CLOCK port. As shown below, FPGA pin AH15 is assigned with 100MHz USER_CLK. We will connect input signal CLOCK to AH15.

Table 1-4: **Oscillator Socket Connections**

Reference Designator	Clock Name	FPGA Pin	Description
X1	USER_CLK	AH15	100 MHz single-ended
U8	CLK_33MHZ_FPGA	AH17	33 MHz single-ended
U8	CLK_27MHZ_FPGA	AG18	27 MHz single-ended
U8	CLK_FPGA_P	L19	200 MHz differential pair (pos)
U8	CLK_FPGA_N	K19	200 MHz differential pair (neg)

Lab1 C

In project 3, we build up a Jackpot Game. The LEDs are turned on one a time in a time sequential manner. Assign each a DIP switch to each of the LEDs. At any point in time, if a switch is turned on corresponding to the glowing LED, all the LEDs start glowing.

1. Comparator

   When the input of DIP is exactly equal to the glowing LEDS, the light will be glowing. So we need a comparator to identity if the input signals are equal to output LEDS. Also, we use WIN_FLAG to indicate that weather the user win the game.

   The pseudo code is shown below:

   If (RESET) WIN_FLAG = 0;

   Else if (LEDS == SWITCHES) WIN_FLAG = 1;

   Else WIN_FLAG = WIN_FLAG;

2. Output LEDS

   The clock for the One-Hot fashion is almost the same as the modified clock in lab1 b. The only difference is that the counter count to a smaller number instead of 5e7, since we need a faster glowing speed. The modified clock frequency is set to 2Hz in this project.

   If the WIN_FLAG is inactivated, the output remains the one-how fashion state. If the WIN_FLAG is activated, all the outputs turn to 1 at the same time.

The pseudo code is shown below:

```
If (RESET)              OUT <= 1;
Else if (WIN_FLAG == 1)  OUT = 1111;
Else                     OUT [3:0] = Shifted OUT [3:0];
```

3. Results

Lab1 a

When run the code on FPGA, all the four LEDs are turned off at the beginning. When set DIP switch to high, the corresponding LED turn on. All four LEDs are tested on the development board.

Lab1 b

When run the code on FPGA, all the four LEDs are turned off at the beginning. When the up button is pressed, the LEDs will count up. When the down button is pressed, the LEDs will count down. If the button is released during the counting, the LEDs will stay the same. If reset button is pressed, all four LEDs are turned off.

Lab1 c

When run the code on FPGA, the four LEDs are glowing one by one at the frequency of 2Hz. If a button is pressed when the corresponding LED is turned on, all the four LEDs start glowing. However, if a button is pressed but the corresponding LED is not turned on; the LEDs will remain state glowing one by one.

4. Conclusion

It is very easy to implement a logic circuit on a FPGA development board. Both the design time and the test time are saved during the design.

A counter is very useful in the FPGA design, since the user clock is set to a constant value. A counter can generate different frequency clock for different design.

The state of the circuit should respond to trigger, like posedge. Or the logic of the circuit might mess up.

5. Questions

(a) How are the user push-buttons wired on the XUP board?
The user push-buttons are active high. Hence, when the button is pressed, the output of the button is pull up.

Reference Designator	Label/Definition	FPGA Pin
SW10	N(GPIO North)	U8
SW11	S(GPIO South)	V8
SW12	E(GPIO East)	AK7
SW13	W(GPIO West)	AJ7
SW14	C(GPIO Center)	Aj6

(b) What is the purpose of an edge detection circuit and how should it have been used in this lab?

The purpose of an edge detection circuit is to detect the edge of a signal when the signal is transferred from 0 to 1 or from 1 to 0. When detect the edge, the circuit could run a part of the code for 1 time. However, if there is no edge detection circuit, the code will keep running as long as the state is remained. Just like the push button in the lab1 b. When the up or down button is pressed, the counter will keep counting up or down.

In lab2 c, if and only if a DIP switch is pressed (trigger is detected), the comparator will compare the DIP switch input and the LEDs output to decide if the user is win.

6. Appendix

Lab1 a.

Source code:

```
/*****************************************/
`timescale 1ns / 1ps
/*****************************************/
// Create Date:    22:54:09 09/04/2013
// Module Name:    switch
/*****************************************/
module switch(SWITCHES,LEDS);
   input      [3:0]    SWITCHES;      // 4-bits input switches
   output    [3:0] LEDS;                        // 4-bits output leds
   assign     LEDS[3:0] = SWITCHES[3:0]; // assigne the value of switches to the leds
endmodule
/*****************************************/
```

.ucf file:

```
/*****************************************/
NET "SWITCHES[0]"        LOC = "U25";
NET "SWITCHES[1]"        LOC = "AG27";
NET "SWITCHES[2]"        LOC = "AF25";
NET "SWITCHES[3]"        LOC = "AF26";

NET "LEDS[0]"              LOC = "H18";
NET "LEDS[1]"              LOC = "L18";
NET "LEDS[2]"              LOC = "G15";
NET "LEDS[3]"              LOC = "AD26";
/*****************************************/
```

Lab1 b

Source Code:

```
/*****************************************/

`timescale 1ns / 1ps

/*****************************************/
// Create Date:    12:37:01 09/05/2013
// Module Name:    led_count
/*****************************************/
module led_count(
   input RESET,
   input [1:0] I_SWITCHES,
   input CLOCK,
   output [3:0] O_LEDS
   );
/*****************************************/
      wire W_CLK_1HZ;                  // 1Hz clock
      reg  R_CLK_1HZ;                  // register for 1Hz clock
      reg  [29:0] R_COUNT;             // counter to generate 1 Hz clock
      reg  [3:0] R_OUT;                // output data register
      assign O_LEDS[3:0] = R_OUT[3:0]; // assign the output register to the leds
      assign W_CLK_1HZ = R_CLK_1HZ;    // assign the clock register to the clock wire
/*****************************************/
      always @(posedge     CLOCK)      // the original 100MHz clock is divided by 10^8
            begin                      // generate the 1Hz clock R_CLK_1Hz
            if (R_COUNT == 50000000)   // if R_COUNT =  5e7, invert R_CLK_1HZ, R_COUNT
back to zero
                  begin
                  R_COUNT <= 0;
                  R_CLK_1HZ <= !R_CLK_1HZ;
                  end
            else                       // if R_COUNT <= 5e7, R_COUNT counts up,
R_CLK_1HZ remain
                  begin
                  R_COUNT <= R_COUNT + 1;
                  R_CLK_1HZ <= R_CLK_1HZ;
```

```
                end
        end
/*****************************************/
        always @(posedge W_CLK_1HZ or posedge RESET) // switch control part
                begin
                        if(RESET)                       // if RESET = 1, R_OUT becomes zero
                                R_OUT <= 0;
                        else if(I_SWITCHES[1])          // if down_botton is pressed, R_OUT
substracted one by one
                                R_OUT <= R_OUT - 1;
                        else if(I_SWITCHES[0])          // if up_botton is pressed, R_OUT added
one by one
                                R_OUT <= R_OUT + 1;
                end

endmodule
/*****************************************/
```

.ucf file:

```
/*****************************************/
NET "I_SWITCHES[0]"     LOC = "U8";
NET "I_SWITCHES[1]"     LOC = "V8";
NET "RESET"             LOC = "AJ6";

NET "O_LEDS[3]"         LOC = "H18";
NET "O_LEDS[2]"         LOC = "L18";
NET "O_LEDS[1]"         LOC = "G15";
NET "O_LEDS[0]"         LOC = "AD26";
NET "CLOCK"             LOC = "AH15";
NET "CLOCK"       TNM_NET = "CLOCK";
TIMESPEC  "TS_CLOCK" = PERIOD "CLOCK"  10ns  HIGH  50%;
/*****************************************/
```

Lab1 c

Source Code:

```
/*****************************************/
`timescale 1ns / 1ps
//////////////////////////////////////////
// Create Date:    20:18:12 09/09/2013
// Module Name:    jackpot
//////////////////////////////////////////
module jackpot(
   input [3:0] I_SWITCHES,          // three inputs are I_SWITCHES, I_RESET, I_CLK
   input I_RESET,                           // one output is I_LEDS
   input I_CLK,
   output [3:0] I_LEDS
   );

      wire W_CLK_2HZ;                  // wire for frequency-modified clock
      reg      R_CLK_2HZ;              // register for frequency-modified clock
      reg      [29:0] R_CNT;           // register for the counter to modify the clock
frequency
      reg      [3:0] R_OUT;            // register for output
      reg      R_WIN_FLAG;             // register for win flag

      assign I_LEDS[3:0] = R_OUT[3:0];      // assign register R_OUT to output I_LEDS
      assign SWITCH_FLAG = I_SWITCHES[0] | I_SWITCHES[1] | I_SWITCHES[2] |
I_SWITCHES[3];              // if any switches input is 1, the SWITCH_FLAG is high
      assign W_CLK_2HZ = R_CLK_2HZ;     // assign register R_CLK_2HZ to wire W_CLK_2HZ
/*****************************************/
      always @(posedge     I_CLK or posedge I_RESET)  // the original 100MHz clock is divided
by 2.5e7
            begin                      // generate the 2Hz clock R_CLK_2HZ
            if(I_RESET)                // if I_RESET input is active, R_CNT back to zero,
R_CLK_2HZ becomes low
             begin
                  R_CNT <= 0;
                  R_CLK_2HZ <= 0;
            end
```

```
            else
             begin
                    if (R_CNT == 25000000)                  // if R_CNT =  5e7, invert R_CLK_2HZ,
R_CNT back to zero
                     begin
                            R_CNT <= 0;
                            R_CLK_2HZ <= ~R_CLK_2HZ;
                     end
                    else
                     begin
    // if R_COUNT <= 2.5e7, R_CNT counts up
                            R_CNT <= R_CNT + 1;
                     end
             end
     end
/*****************************************/
     always @(posedge W_CLK_2HZ or posedge I_RESET) // LEDS control part
            begin
                    if(I_RESET)
            // if I_RESET is active, the output is 0001
                            R_OUT <= 1;
                    else
                    begin
                            if(R_WIN_FLAG == 1)
    // if R_WIN_FLAG is active, output is 1111
                                    R_OUT <= 4'b1111;
                            else
                    // if I_RESET=0 and R_WIN_FLAG=0, R_OUT shift one by one
                                    R_OUT[3:0] <= {R_OUT[0], R_OUT[3:1]};
                    end
            end
/*****************************************/
     always @(posedge SWITCH_FLAG or posedge I_RESET) //Comparator part
            begin
                    if(I_RESET)                 // if I_RESET is active, win flag is 0
                            R_WIN_FLAG <= 0;
                    else if (I_LEDS == I_SWITCHES)          // if I_LEDS = I_SWITCHES, win flag is
active
```

```
                        R_WIN_FLAG <= 1;
                else            // else win flag remain
                        R_WIN_FLAG <= R_WIN_FLAG;
        end

endmodule
/*****************************************/
```

.ucf file:

```
/*****************************************/
NET "I_SWITCHES[0]"     LOC = "U25";
NET "I_SWITCHES[1]"     LOC = "AG27";
NET "I_SWITCHES[2]"     LOC = "AF25";
NET "I_SWITCHES[3]"     LOC = "AF26";
NET "I_RESET"           LOC = "AJ6";


NET "I_LEDS[0]"         LOC = "H18";
NET "I_LEDS[1]"         LOC = "L18";
NET "I_LEDS[2]"         LOC = "G15";
NET "I_LEDS[3]"         LOC = "AD26";
NET "I_CLK"             LOC = "AH15";
NET "I_CLK"       TNM_NET = "I_CLK";

TIMESPEC  "TS_CLOCK" = PERIOD "I_CLK"  10ns  HIGH  50%;
/*****************************************/
```